UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/748,165	12/27/2000	Nigel C. Paver	884.A96US1	5048

45457 7590 04/15/2008
SCHWEGMAN, LUNDBERG & WOESSNER/Intel
PO BOX 2938
MINNEAPOLIS, MN 55402

EXAMINER	
HUISMAN, DAVID J	
ART UNIT	**PAPER NUMBER**
2183	

MAIL DATE	DELIVERY MODE
04/15/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	**Application No.** 09/748,165	**Applicant(s)** PAVER, NIGEL C.	
	Examiner DAVID J. HUISMAN	**Art Unit** 2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on 12 November 2007.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 1-19 and 23-25 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 1-19 and 23-25 is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on 05 October 2007 is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)	4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)	5) ☐ Notice of Informal Patent Application
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.	6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-19 and 23-25 have been examined.

Papers Submitted

2. It is hereby acknowledged that the following papers have been received and placed of record in the file: Power of Attorney as received on 10/7/2005, Power of Attorney as received on 3/26/2007, Petition to Revive Application (which was ultimately granted on 2/27/2008) After-Final Amendment, Extension of Time, and RCE as received on 10/5/2007, and Supplemental Amendment as received on 11/12/2007.

Specification

3. The amended title of the invention is not descriptive. A new title is required that is clearly indicative of the invention to which the claims are directed. The examiner recommends amending the title to more accurately reflect the content of the independent claims, i.e., the combination of arithmetic flags.

Claim Objections

4. Claim 12 is objected to because of the following informalities: In line 2, insert a comma before “when” and a comma before “result”. Appropriate correction is required.

Claim Rejections - 35 USC § 112

5. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

6. Claims 1-6 and 23-25 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

7. Claim 1 recites the limitation "the word" in at least line 6. There is insufficient antecedent basis for this limitation in the claim because it is not clear if applicant is referring to "an N-bit word" in line 3 or "a word" in line 4.

8. Claim 23 recites the limitation "the word" in at least line 6. There is insufficient antecedent basis for this limitation in the claim because it is not clear if applicant is referring to "an N-bit word" in line 3 or "a word" in line 4.

9. Claims 2-6 and 24-25 are rejected under 35 U.S.C 112, 2nd paragraph, for being indefinite, because they are dependent, either directly or indirectly, on an indefinite claim.

Claim Rejections - 35 USC § 102

10. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

11. Claims 1-4, 6-9, 17-19, and 23 are rejected under 35 U.S.C. 102(e) as being anticipated by Wilson, U.S. Patent No. 6,530,012.

12. Referring to claim 1, Wilson has taught a device to manage a plurality of arithmetic flags, wherein an M-bit set of arithmetic flags (see Fig.5, and note CCX0 to CCX7 make up a 32-bit set of flags) is associated with each of a plurality of data items of varying field sizes within an N-bit word (see column 1, lines 17-22, and column 4, lines 20-24; within a 64-bit word/operand, there are various field sizes (byte, halfword, word, longword), M and N being positive integers, comprising a combination function module (Fig.6) to:

a) examine a word comprising a plurality of sets of arithmetic flags. See column 6, line 46, to column 7, line 6, and note that the processor is to set certain condition code values to values of other condition codes. For instance, in the case of halfword, CCX0 would be set and then CCX1 would be set equal to the value in CCX) (or vice versa). In order to perform this copy, the system must "examine" the contents of one condition code field so that it knows how to set the other condition code field.

b) determine a data item field size for the word, and based on the determination of the data item field size to logically combine the plurality of arithmetic flags of the sets within the word into a single combined arithmetic flag variable of M bits, wherein the plurality of arithmetic flags represent a result status of the plurality of data items after a mathematical operation is performed by a processor on the plurality of data items. It should be realized that a field size must first be determined. For purposes of this examination, "field size" in Wilson refers to the maximum number of unique condition codes that can be produced for a given instruction. A condition code is a 4-bit set of arithmetic flags (column 6, lines 37-42). From column 6, lines 43-45, when the SIMD data is one byte in size, it is determined that the field size is 8. More specifically, up to 8 unique condition codes will be produced. From column 6, lines 46-56, when the SIMD data is two bytes (half-word) in size, it is determined that the field size is 4. That is, 4 condition codes are produced and then duplicated. However, it is only possible for 4 unique codes to be produced. As a further example, from column 6, lines 58-65, when the SIMD data is 32 bits in size (word), the field size is determined to be 2. That is, 2 condition codes are produced and then duplicated multiple times. However, it is only possible for 2 unique codes to be produced. Finally, when the SIMD data size is 64 bits, the field size is 1, where 1 condition code is produced and duplicated multiple times. However, it is only possible for a single unique code to be produced. Once the field size is determined, the flags are logically combined by setting each code according to column 6, line 43, to column 7, line 6, and then storing them together in a single 64-bit arithmetic flag register (see Fig.5). These flags represent result status (column 6, lines 1-5) after a mathematical operation is performed by the processor.

13. Referring to claim 2, Wilson has taught the device recited in claim 1, further comprising a condition check module that determines the result status of the combined arithmetic flag variable and causes the processor to execute an appropriate operation based on the status. See column 8, lines 23-61.

14. Referring to claim 3, Wilson has taught the device recited in claim 1, wherein the field size is based on either a nibble, byte, half-word, or word in length. As described in the rejection of claim 1 above, and in column 6, lines 43-65, the field size is either a byte, half-word, word, or long-word. And, due to the applicant's use of the word "or" in the claim, Wilson is able to anticipate the claim since at least one of the claimed features has been taught.

15. Referring to claim 4, Wilson has taught the device recited in claim 3, wherein the plurality of arithmetic flags further comprise a negative data value, a zero data value, a carry out

occurrence in a data value, or an overflow condition in a data item in the plurality of data items. See column 6, lines 1-5.

16. Referring to claim 6, Wilson has taught the device recited in claim 2, wherein the result status determined by the condition further comprises:

a) any data item has overflowed. If any one of the condition codes in the combined variable of Fig.7 has the V bit set (which represents overflow), then it is determined, by the CC Checker (Fig.7, component 50), that any data item has overflowed. See Table 1 in column 6 for further information.

b) any data item has not overflowed. If any one of the condition codes in the combined variable of Fig.7 has the V bit cleared (which represents no overflow), then it is determined, by the CC Checker (Fig.7, component 50), that any data item has not overflowed. See Table 1 in column 6 for further information.

c) any data item is positive or zero. If any one of the condition codes in the combined variable of Fig.7 has the N bit cleared (which represents a number not being negative), then it is determined, by the CC Checker (Fig.7, component 50), that any data item is positive or zero. See Table 1 in column 6 for further information.

d) any data item is negative. If any one of the condition codes in the combined variable of Fig.7 has the N bit set (which represents a negative number), then it is determined, by the CC Checker (Fig.7, component 50), that any data item is negative. See Table 1 in column 6 for further information.

e) any data item is zero. If any one of the condition codes in the combined variable of Fig.7 has the Z bit set (which represents zero), then it is determined, by the CC Checker (Fig.7, component 50), that any data item is zero. See Table 1 in column 6 for further information.

f) any data item is not zero. If any one of the condition codes in the combined variable of Fig.7 has the Z bit cleared (which represents a non-zero number), then it is determined, by the CC Checker (Fig.7, component 50), that any data item is not zero. See Table 1 in column 6 for further information.

g) any data item has a carry out. If any one of the condition codes in the combined variable of Fig.7 has the C bit set (which represents a carry), then it is determined, by the CC Checker (Fig.7, component 50), that any data item has a carry out. See Table 1 in column 6 for further information.

h) any data item does not have a carry out. If any one of the condition codes in the combined variable of Fig.7 has the C bit cleared (which represents no carry), then it is determined, by the CC Checker (Fig.7, component 50), that any data item does not have a carry out. See Table 1 in column 6 for further information.

i) all data items have overflowed. If all of the condition codes in the combined variable of Fig.7 has the V bit set (which represents overflow), then it is determined, by the CC Checker (Fig.7, component 50), that all data items have overflowed. See Table 1 in column 6 for further information.

j) all data items have not overflowed. If all of the condition codes in the combined variable of Fig.7 has the V bit cleared (which represents no overflow), then it is determined, by the CC

Checker (Fig.7, component 50), that all data items have not overflowed. See Table 1 in column 6 for further information.

k) all data items are positive or zero. If all of the condition codes in the combined variable of Fig.7 has the N bit cleared (which represents a number not being negative), then it is determined, by the CC Checker (Fig.7, component 50), that all data items are positive or zero. See Table 1 in column 6 for further information.

l) all data items are negative. If all of the condition codes in the combined variable of Fig.7 has the N bit set (which represents a negative number), then it is determined, by the CC Checker (Fig.7, component 50), that all data items are negative. See Table 1 in column 6 for further information.

m) all data items are zero. If all of the condition codes in the combined variable of Fig.7 has the Z bit set (which represents zero), then it is determined, by the CC Checker (Fig.7, component 50), that all data items are zero. See Table 1 in column 6 for further information.

n) all data items are not zero. If all of the condition codes in the combined variable of Fig.7 has the Z bit cleared (which represents a non-zero number), then it is determined, by the CC Checker (Fig.7, component 50), that all data items are not zero. See Table 1 in column 6 for further information.

o) all data items have a carry out. If all of the condition codes in the combined variable of Fig.7 has the C bit set (which represents a carry), then it is determined, by the CC Checker (Fig.7, component 50), that all data items have a carry out. See Table 1 in column 6 for further information.

p) all data items do not have a carry out. If all of the condition codes in the combined variable of Fig.7 has the C bit cleared (which represents no carry), then it is determined, by the CC Checker (Fig.7, component 50), that all data items do not have a carry out. See Table 1 in column 6 for further information.

17. Referring to claim 7, Wilson has taught a method of combining a plurality of arithmetic flags for presentation to a processor, comprising:

a) determining a field size of the plurality of arithmetic flags on which to base a combination process, wherein the plurality of arithmetic flags represent a result status of a plurality of data items after a mathematical operation is performed by the processor on the plurality of data items. It should be realized that a field size must first be determined. For purposes of this examination, “field size” in Wilson refers to the maximum number of unique condition codes that can be produced for a given instruction. A condition code is a 4-bit set of arithmetic flags (column 6, lines 37-42). From column 6, lines 43-45, when the SIMD data is one byte in size, it is determined that the field size is 8. More specifically, up to 8 unique condition codes will be produced. From column 6, lines 46-56, when the SIMD data is two bytes (half-word) in size, it is determined that the field size is 4. That is, 4 condition codes are produced and then duplicated. However, it is only possible for 4 unique codes to be produced. As a further example, from column 6, lines 58-65, when the SIMD data is 32 bits in size (word), the field size is determined to be 2. That is, 2 condition codes are produced and then duplicated multiple times. However, it is only possible for 2 unique codes to be produced. Finally, when the SIMD data size is 64 bits, the field size is 1, where 1 condition code is produced and duplicated multiple times. However, it is only possible for a single unique code to be produced. These condition codes (arithmetic flags) represent data item status (column 6, lines 1-5) after a mathematical operation is performed by the processor.

b) extracting the plurality of arithmetic flags based on the field size. The "American Heritage® Dictionary of the English Language, Third Edition," 1992, has defined "extract" as "to derive or obtain from a source" and "to determine or calculate." As a result, Wilson's system will extract (derive, obtain, or determine) the arithmetic flag values based on a mathematical operation performed by the processor. And, from the rejection of claim 7(a) above, the number of flags derived (extracted) depends on the field size.

c) combining the plurality of arithmetic flags based on a function selected when a combination process is selected. See Fig.6 and column 6, line 43, to column 7, line 6. Note that the functions would include the byte combination, the half-word combination, the word combination, and the long-word combination.

d) storing a result of the combining of the plurality of arithmetic flags in a destination register for access by the processor. Note that the flags are stored in the destination register shown at the bottom of Fig.6.

18. Referring to claim 8, Wilson has taught a method as described in claim 7. Furthermore, claim 8 is rejected for the same reasons set forth in the rejection of claim 3.

19. Referring to claim 9, Wilson has taught a method as described in claim 8. Furthermore, claim 9 is rejected for the same reasons set forth in the rejection of claim 4.

20. Referring to claim 12, Wilson has taught an apparatus comprising a data storage medium for storing instructions (see Fig.1), wherein the instructions when executed by a processor result in the processor performing a method, the method comprising:

a) determining a field size of the plurality of arithmetic flags on which to base a combination process, wherein the plurality of arithmetic flags represent the status of a plurality of data items after a mathematical operation is performed by the processor on the plurality of data items. It should be realized that a field size must first be determined. For purposes of this examination, "field size" in Wilson refers to the maximum number of unique condition codes that can be produced for a given instruction. A condition code is a 4-bit set of arithmetic flags (column 6, lines 37-42). From column 6, lines 43-45, when the SIMD data is one byte in size, it is determined that the field size is 8. More specifically, up to 8 unique condition codes will be produced. From column 6, lines 46-56, when the SIMD data is two bytes (half-word) in size, it is determined that the field size is 4. That is, 4 condition codes are produced and then duplicated. However, it is only possible for 4 unique codes to be produced. As a further example, from column 6, lines 58-65, when the SIMD data is 32 bits in size (word), the field size is determined to be 2. That is, 2 condition codes are produced and then duplicated multiple times. However, it is only possible for 2 unique codes to be produced. Finally, when the SIMD data size is 64 bits, the field size is 1, where 1 condition code is produced and duplicated multiple times. However, it is only possible for a single unique code to be produced. These condition codes (arithmetic flags) represent data item status (column 6, lines 1-5) after a mathematical operation is performed by the processor.

b) extracting the plurality of arithmetic flags based on the field size. The "American Heritage® Dictionary of the English Language, Third Edition," 1992, has defined "extract" as "to derive or obtain from a source" and "to determine or calculate." As a result, Wilson's system will extract (derive, obtain, or determine) the arithmetic flag values based on a mathematical operation performed by the processor. And, from the rejection of claim 12(a) above, the number of flags derived (extracted) depends on the field size.

c) combining the plurality of arithmetic flags based on a function selected when a combination process is selected. See Fig.6 and column 6, line 43, to column 7, line 6. Note that the functions would include the byte combination, the half-word combination, the word combination, and the long-word combination.

d) storing a result of the combining of the plurality of arithmetic flags in a destination register for access by the processor. Note that the flags are stored in the destination register shown at the bottom of Fig.6.

21. Referring to claim 13, Wilson has taught an apparatus as described in claim 12. Furthermore, claim 13 is rejected for the same reasons set forth in the rejection of claim 3.

22. Referring to claim 14, Wilson has taught an apparatus as described in claim 13. Furthermore, claim 14 is rejected for the same reasons set forth in the rejection of claim 4.

23. Referring to claim 17, Wilson has taught a method of extracting a plurality of arithmetic flags for presentation to a processor, comprising:

a) determining a field size of the plurality of arithmetic flags on which to base a combination process, wherein the plurality of arithmetic flags represent a result status of a plurality of data items after a mathematical operation is performed by the processor on the plurality of data items. It should be realized that a field size must first be determined. For purposes of this examination, “field size” in Wilson refers to the maximum number of unique condition codes that can be produced for a given instruction. A condition code is a 4-bit set of arithmetic flags (column 6, lines 37-42). From column 6, lines 43-45, when the SIMD data is one byte in size, it is determined that the field size is 8. More specifically, up to 8 unique condition codes will be produced. From column 6, lines 46-56, when the SIMD data is two bytes (half-word) in size, it is determined that the field size is 4. That is, 4 condition codes are produced and then duplicated. However, it is only possible for 4 unique codes to be produced. As a further example, from column 6, lines 58-65, when the SIMD data is 32 bits in size (word), the field size is determined to be 2. That is, 2 condition codes are produced and then duplicated multiple times. However, it is only possible for 2 unique codes to be produced. Finally, when the SIMD data size is 64 bits, the field size is 1, where 1 condition code is produced and duplicated multiple times. However, it is only possible for a single unique code to be produced. These condition codes (arithmetic flags) represent data item status (column 6, lines 1-5) after a mathematical operation is performed by the processor.

b) extracting the plurality of arithmetic flags based on the field size. The "American Heritage® Dictionary of the English Language, Third Edition," 1992, has defined "extract" as "to derive or obtain from a source" and "to determine or calculate." As a result, Wilson's system will extract (derive, obtain, or determine) the arithmetic flag values based on a mathematical operation performed by the processor. And, from the rejection of claim 17(a) above, the number of flags derived (extracted) depends on the field size.

c) storing a result of the extracting of the plurality of arithmetic flags in a destination register for access by the processor. Note that the flags are stored in the destination register shown at the bottom of Fig.6.

24. Referring to claim 18, Wilson has taught a method as described in claim 17. Furthermore, claim 18 is rejected for the same reasons set forth in the rejection of claim 3.

25. Referring to claim 19, Wilson has taught a method as described in claim 18. Furthermore, claim 19 is rejected for the same reasons set forth in the rejection of claim 4.

26. Referring to claim 23, Wilson has taught a system to manage a plurality of arithmetic flags, wherein an M-bit set of arithmetic flags (see Fig.5, and note CCX0 to CCX7 make up a 32-bit set of flags) is associated with each of a plurality of data items of varying field sizes within an N-bit word (see column 1, lines 17-22, and column 4, lines 20-24; within a 64-bit word/operand, there are various field sizes (byte, halfword, word, longword), M and N being positive integers, comprising a combination function module (Fig.6) to:

a) examine a word comprising a plurality of sets of arithmetic flags. See column 6, line 46, to column 7, line 6, and note that the processor is to set certain condition code values to values of other condition codes. For instance, in the case of halfword, CCX0 would be set and then CCX1 would be set equal to the value in CCX) (or vice versa). In order to perform this copy, the system must "examine" the contents of one condition code field so that it knows how to set the other condition code field.

b) determine a data item field size for the word, and based on the determination of the data item field size to logically combine the plurality of arithmetic flags of the sets within the word into a single combined arithmetic flag variable of M bits, wherein the plurality of arithmetic flags represent a result status of the plurality of data items after a mathematical operation is performed by a processor on the plurality of data items. It should be realized that a field size must first be determined. For purposes of this examination, "field size" in Wilson refers to the maximum number of unique condition codes that can be produced for a given instruction. A condition code is a 4-bit set of arithmetic flags (column 6, lines 37-42). From column 6, lines 43-45, when the

SIMD data is one byte in size, it is determined that the field size is 8. More specifically, up to 8 unique condition codes will be produced. From column 6, lines 46-56, when the SIMD data is two bytes (half-word) in size, it is determined that the field size is 4. That is, 4 condition codes are produced and then duplicated. However, it is only possible for 4 unique codes to be produced. As a further example, from column 6, lines 58-65, when the SIMD data is 32 bits in size (word), the field size is determined to be 2. That is, 2 condition codes are produced and then duplicated multiple times. However, it is only possible for 2 unique codes to be produced. Finally, when the SIMD data size is 64 bits, the field size is 1, where 1 condition code is produced and duplicated multiple times. However, it is only possible for a single unique code to be produced. Once the field size is determined, the flags are logically combined by setting each code according to column 6, line 43, to column 7, line 6, and then storing them together in a single 64-bit arithmetic flag register (see Fig.5). These flags represent result status (column 6, lines 1-5) after a mathematical operation is performed by the processor.

c) a processor including a condition check module coupled to the combination function module, the processor to receive the single combined arithmetic flag variable and to determine the next operation to perform based upon the result status of the single combined arithmetic flag variable. See column 8, lines 23-61.

Claim Rejections - 35 USC § 103

27. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person

having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

28. Claims 5, 10-11, and 15-16 are rejected under 35 U.S.C. 103(a) as being unpatentable over Wilson in view of Bindloss et al., U.S. Patent No. 5,778,241 (herein referred to as Bindloss).

29. Referring to claim 5, Wilson has taught the device recited in claim 4. Wilson has not taught that the combination function module performs either an AND or an OR operation. However, Bindloss has taught the general concept of ANDing multiple condition codes together to produce an overall condition code. See Fig.3C and column 7, lines 32-36. Performing this operation would allow the system, for instance, to check if all data items have a carry. See column 4, lines 1-4. This should be realized because if 4 additions occur, each addition producing condition codes of 1101, 1011, 1100, and 1110, respectively (where the carry flag is the 0th bit), then by ANDing each code together, the final code is 1000. And, since the 0th bit is equal to 0, the system would recognize that a carry did not occur in all of the items. A person of ordinary skill in the art would have recognized that checking a single ANDed condition code would take less time than checking multiple condition codes. Therefore, to increase the efficiency of the system, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Wilson such that the combination function module performs either an AND or an OR operation. Note that Bindloss' teaching of just an AND operation is sufficient enough to reject the claim since the applicant is using alternate language ("or").

Under a second interpretation, the examiner asserts that components within a processor are known to include AND or OR gates since these gates are some of the most basic of all building block logic elements. Hence, it would have been obvious to one of ordinary skill in the art to modify Wilson's combination function module to include at least one AND or OR gate so that such an operation is performed.

30. Referring to claim 10, Wilson has taught a method as described in claim 9. Furthermore, claim 10 is rejected for the same reasons set forth in the rejection of claim 5.

31. Referring to claim 11, Wilson in view of Bindloss has taught a method as described in claim 10. Furthermore, claim 11 is rejected for the same reasons set forth in the rejection of claim 6.

32. Referring to claim 15, Wilson has taught an apparatus as described in claim 14. Furthermore, claim 15 is rejected for the same reasons set forth in the rejection of claim 5.

33. Referring to claim 16, Wilson has taught an apparatus as described in claim 15. Furthermore, claim 16 is rejected for the same reasons set forth in the rejection of claim 6.

34. Claims 24-25 are rejected under 35 U.S.C. 103(a) as being unpatentable over Wilson in view of the examiner's taking of Official Notice.

35. Referring to claim 24, Wilson has taught the system of claim 23. Wilson has not explicitly taught that the processor includes at least three stages of pipelining. However, Official Notice is taken that pipelining into at least three stages is well known and advantageous in the art. Specifically, N-stage pipelining allows for the overlapping of execution of N instructions, thereby increasing throughput by a factor of N. That is, without pipelining, instructions are executed serially. If each instruction takes X cycles to complete, then to execute M instructions, MX cycles would be required. However, with pipelining when a first instruction exits the first stage and enters a second stage, a new instruction may enter the first stage behind it. This allows

for each instruction to complete in roughly once cycle. So, M instructions would require much less than MX cycles to execute. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Wilson to include at least three stages of pipelining.

36. Referring to claim 25, Wilson, as modified, has taught the system of claim 24. Wilson has not taught that the at least three stages of pipelining include a fetch stage, a decode stage, and an execute stage. However, Official Notice is taken that pipeline stages including these operations are well known in the art. That is, it would be well known to modify the fetch, decode, and execute circuitry of Fig.1 of Wilson to be pipelined so that instructions may be fetched, decoded, and executed in parallel, thereby increasing throughput. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Wilson such that the at least three stages of pipelining include a fetch stage, a decode stage, and an execute stage.

Response to Arguments

37. Applicant's arguments filed on November 12, 2007, have been fully considered but they are not persuasive.

38. Applicant argues the novelty/rejection of claim 1 on page 15 of the remarks, in substance that:

> "There is no disclosure in Wilson of "logically combine the plurality of arithmetic flags of the sets within the word into a single combined arithmetic flag variable of M bits, wherein the plurality of arithmetic flags represent a result status of the plurality of data items after a mathematical operation is performed by the processor on the plurality of data items", as recited in independent claims 1 and 23."

39. These arguments are not found persuasive for the following reasons:

a) The logical combination of flags clearly occurs by storing each group of flags adjacent to one another in a single register. See Fig.5. This is a logical combination, as bits having logical values are combined to form a group of bits. If applicant performs a logical combination in a different manner, then that specific way should be claimed.

40. Applicant argues the novelty/rejection of claims 6, 11, and 16 on page 18 of the remarks, in substance that:

> "…these bytes in test register Treg, referred to in Table 1 of Wilson, are programmable and set to the required values prior to instruction execution. However, there is no disclosure in Wilson that teaches a result status associated with a plurality of data items after a mathematic operation is performed by the processor on the plurality of data item, as recited in claims 6, 11, and 16. Therefore, Wilson fails to disclose these additional elements and so fails to anticipate claims 6, 11, and 16."

41. These arguments are not found persuasive for the following reasons:

a) While applicant is correct that the data in Treg are predetermined, the examiner is not interpreting Treg as the result status of mathematical operations. The examiner is using the condition codes (Fig.5) as the result status. Fig.7 shows that these are distinct components. The condition code values include arithmetic flags found in Treg however. See Fig.5. They are set based on mathematical operations. Then, for conditional execution, the system could compare the derived condition codes with the preset Treg value (as discussed in column 8, lines 23-61).

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to DAVID J. HUISMAN whose telephone number is (571)272-4168. The examiner can normally be reached on Monday-Friday (8:00-4:30).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/David J. Huisman/
Primary Examiner, Art Unit 2183
April 5, 2008